福建师范大学协和学院
物联网工程

大一 C 语言入门题库及知识概要

修订记录

日期	修订版本	修改者	审核人	修改描述
2021-08-05	V1.0	20 物联-林炫	郑中华	首次提交
2023-08-15	V2.0	22 物联兴趣小组	郑中华	第二次修改

物联星辰计划

一、编程规范

1. 命名风格

（1）所有编程相关的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。
说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式也要避免采用。
正例：ali / alibaba / taobao / kaikeba / aliyun / youku / hangzhou 等国际通用的名称，可视同英文。
反例：DaZhePromotion【打折】/ getPingfenByName()【评分】 / String fw【福娃】/ int 变量名 = 3

（2）所有编程相关的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。
反例：_name / __name / $Object / name_ / name$ / Object$

（3）方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase（小驼峰） 风格。
正例：localValue / getHttpMessage() / inputUserId

（4）常量命名应该全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。
正例：MAX_STOCK_COUNT / CACHE_EXPIRED_TIME
反例：MAX_COUNT / EXPIRED_TIME

2. 代码风格

（1）任何二目、三目运算符的左右两边都需要加一个空格。
说明：包括赋值运算符 =、逻辑运算符 &&、加减乘除符号等。
正例：int count = 0 / int sum = num1 + num2
反例：int count=0 / int sum=num1+num2
（2）同一个代码块缩进必须一致
正例：

```
#include <stdio.h>
int main()
{
    printf("欢迎来到物联这个大家庭！\n");
    return 0;
}
```

反例：

物联晨辰计划

```c
#include <stdio.h>
int main()
{
printf("欢迎来到物联这个大家庭！\n");
    return 0;
}
```

（3）大括号对齐

A.规则一：{和}分别都要独占一行。互为一对的{和}要位于同一列，并且与引用它们的语句左对齐。

B.规则二：{}之内的代码要向内缩进一个 Tab，且同一地位要左对齐，地位不同继续缩进。

（4）缩进

缩进是通过键盘上的 Tab 建实现的，缩进可以使程序更有层次感。原则是：如果地位相等不用缩进，如果属于某一个代码的内部就需要缩进。

（5）成对书写

成对的符号一定要成对书写，如 ()、{}。不要写完左括号然后写内容最后在补充右括号，这样容易漏掉右括号，尤其是写嵌套程序的时候。（虽然现在的编译器都是成对的出现，不过还是要注意一下）

（6）代码行

A.规则一：一行代码只做一件事情，如只定义一个变量，或者只写一条语句。这样的代码容易阅读，并且便于注释。

B.规则二：if、else、for、while、do 等语句自占一行，执行语句不得紧跟其后，此外非常重要的一点是：不论执行语句有多少行，就算一行也要加{}，并且遵循对齐的原则，这样可以防止书写失误。

（7）注释

合理的注释有利于提高代码的可读性。

物联星应计划

二、基础知识：

1. 基本数据类型

类型	含义	32位编译器中大小(一般)	64位编译器中大小(一般)	最小值(32位)	最大值(32位)
bool(stdbool.h)	布尔类型	1byte	1byte	false	true
char	单个字符	1byte	1byte	-2^7	2^7-1
short	短整形	2byte	2byte	-2^{15}	$2^{15}-1$
int	整形	4byte	4byte	-2^{31}	$2^{31}-1$
long	长整形	**4byte**	**8byte**	-2^{31}	$2^{31}-1$
long long	长整形	8byte	8byte	-2^{63}	$2^{63}-1$
float	单精度浮点数	4byte	4byte	-2^{127}	2^{128}
double	双精度浮点数	8byte	8byte	-2^{1023}	2^{1024}
long double	扩展精度浮点数	12byte	16byte	-2^{16383}	2^{16384}
char*	字符常量或字符串常量	**4byte**	**8byte**	无意义	无意义

2. 关键字

（1）控制语句关键字（12 个）：

A.循环语句

- for：一种循环语句（可意会不可言传）
- do ：循环语句的循环体
- while ：循环语句的循环条件
- break：跳出当前循环
- continue：结束当前循环，开始下一轮循环

B.条件语句

- if：条件语句
- else ：条件语句否定分支（与 if 连用）
- goto：无条件跳转语句（尽量别使用）

C.开关语句

- switch ：用于开关语句
- case：开关语句分支
- default：开关语句中的“其他”分支

D.返回语句

- return ：子程序返回语句（可以带参数，也可以不带参数）

（2）存储类型关键字（4 个）：

- auto ：声明自动变量 一般不使用
- extern：声明变量是在其他文件正声明（也可以看做是引用变量）
- register：声明寄存器变量
- static ：声明静态变量

（3）其它关键字（8 个）：

- const ：声明只读变量

物联星辰计划

- sizeof：计算数据类型长度
- typedef：用以给数据类型取别名
- volatile：说明变量在程序执行中可被隐含地改变
- unsigned：声明无符号类型变量
- enum：声明枚举类型
- extern：声明变量或函数是在其他文件或本文件的其他位置定义
- struct：声明结构体类型

3. 变量

（1）定义：

```
type variable_list;
```

- **type** 表示变量的数据类型，可以是整型、浮点型、字符型、指针等，也可以是用户自定义的对象。
- **variable_list** 可以由一个或多个变量的名称组成，多个变量之间用逗号（,）分隔，变量由字母、数字和下划线组成，且以字母或下划线开头。

例：`int age;  // age 被定义为一个整形变量`

（2）初始化：

```
type variable_name = value;
```

- type 表示变量的数据类型，variable_name 是变量的名称，value 是变量的初始值。

例：`float pi = 3.14;  // 浮点型变量 pi 初始化为 3.14`

（3）在全局变量和静态变量中（在函数内部定义的静态变量和在函数外部定义的全局变量）：

- 整型变量（int、short、long 等）：默认值为 0。
- 浮点型变量（float、double 等）：默认值为 0.0。
- 字符型变量（char）：默认值为'\0'，即空字符。
- 指针变量：默认值为 NULL，表示指针不指向任何有效的内存地址。
- 数组、结构体、联合等复合类型的变量：它们的元素或成员将按照相应的规则进行默认初始化。

4. 常量

（1）整数常量：

0x 或 0X 表示十六进制，0 表示八进制，不带前缀则默认表示十进制。

- 后缀：U 表示无符号整数（unsigned），L 表示长整数（long）。后缀可以是大写，也可以是小写，U 和 L 的顺序任意。

例：
```
long a = 100000L;
unsigned int b = 10U;
```

（2）浮点数常量：

由整数部分、小数点、小数部分和指数部分组成。带符号的指数是用 e 或 E 引入的，例如：335.7 的指数形式为 3.557e+2

- 浮点数常量可以带有一个后缀表示数据类型，例如 `float m = 3.14f;`

（3）字符常量：

物联星辰计划

'x' x 为一个普通字符
字符常量可以是一个普通的字符(例如'x')、一个转义序列(例如'\ t'),或是一个通用的字符 (例如'\ u02C')
• 转义字符:

\?	在书写连续多个问号是使用，防止他们被解析成三字母词
\'	用于表示字符常量'
\"	用于表示一个字符串内部的双引号
\\	用于表示一个反斜杠，防止它被解释为一个转义序列符
\a	警告字符，蜂鸣
\b	退格符
\f	进纸符（换页）
\n	换行
\r	回车
\t	水平制表符
\v	垂直制表符
\ddd	ddd 表示 1-3 个八进制的数字
\xdd	dd 表示 2 个十六进制数字

（4）字符串常量
字符串字面值或常量是括在双引号""中的，一个字符串包含类似于字符常量的字符:普通字符、转义序列和通用字符。C 语言字符串常量在内存中以终止符\0 结尾。
例："Hello World"

5. 注释方法

（1）多行注释：/*注释内容*/
（2）单行注释：// 注释内容

6. 基本运算符

（1）算术运算符：
+：加法运算符　　-：减法运算符
*：乘法运算符　　/：除法运算符
%：求余运算符（模运算符）　　++：自增运算符
--：自减运算符
（通常情况下，自增自减运算符有以下几种形式：
++a：a 自增 1 后，再取值

物联星辰计划

--a：a 自减 1 后，再取值
a++：a 取值后，再自增 1
a--：a 取值后，再自减 1）

（2）赋值运算符：

=：等号

复合赋值运算符：+=、-=、*=、/=、%=

例：a=a+1;等价于 a+=1;

（3）关系运算符：

>：大于　　>=：大于等于

<：小于　　<=：小于等于

==：等于　　!=：不等于

（4）逻辑运算符：

&&：逻辑与　　||：逻辑或

例：0&&1 结果为 0　　例：0||1 结果为 1

!：逻辑非

例：!0 结果为 1

（5）三目运算符：

?:：条件表达式

- 格式：表达式 1?表达式 2:表达式 3;
- 执行过程：先判断表达式 1 的值是否为真，若为真的话执行表达式 2;反之，执行表达式 3。

（6）指针运算符

*：用于指针变量的解引用，即获取指针变量所指向的值

&：用于获取变量的地址，即获取变量在内存中的位置

7.格式化输出语句

printf("输出格式符", 输出项);

- %d:带符号十进制整数
- %c:单个字符
- %s:字符串
- %f：6 位浮点数

物联星辰计划

三、顺序结构程序设计

1. 顺序结构

顺序结构是程序设计中的一种基本的线性结构。在顺序结构中，程序按照代码的顺序依次执行，就像我们读小说一样，一行一行地看过去。

例题（1）：

输入两个整数，按等式的形式输出它们的和。例如，输入 3 和 2，输出 3+2=5。

//示例：

```c
#include <stdio.h>

int main()
{
    int a = 0;
    int b = 0;
    scanf("%d %d",&a, &b);
    int sum = a + b;
    printf("The sum of %d and %d is %d\n", a, b, sum);
    return 0;
}
```

代码执行顺序为：定义整数变量 a、b -> 键盘输入数据分别赋值给 a、b -> 定义整数变量 sum,并初始化为 a、b 之和 -> 屏幕输出字符串

例题（2）：

输入秒数，将它化成小时、分、秒的格式输出，例如输入 7278 秒，出 02:01:18。注意：若输入 3500 秒，则只能输出 00:58:20。

```c
//逐步求解示例：
#include <stdio.h>

int main()
{
    int time_Data;
    scanf("%d", &time_Data);
    int hour = 0, minute = 0, second = 0;
    hour = time_Data / 60 / 60;//提取小时
    minute = time_Data / 60 % 60;//提取分钟
    second = time_Data % 60 % 60;//提取分秒

    printf("%02d:%02d:%02d", hour, minute, second);
    return 0;
}
```

解题思路->理解好题目要求，运用时分秒进行结果的输出，可逐步进行时分秒求解，也可在原有基础上继续往下求解。

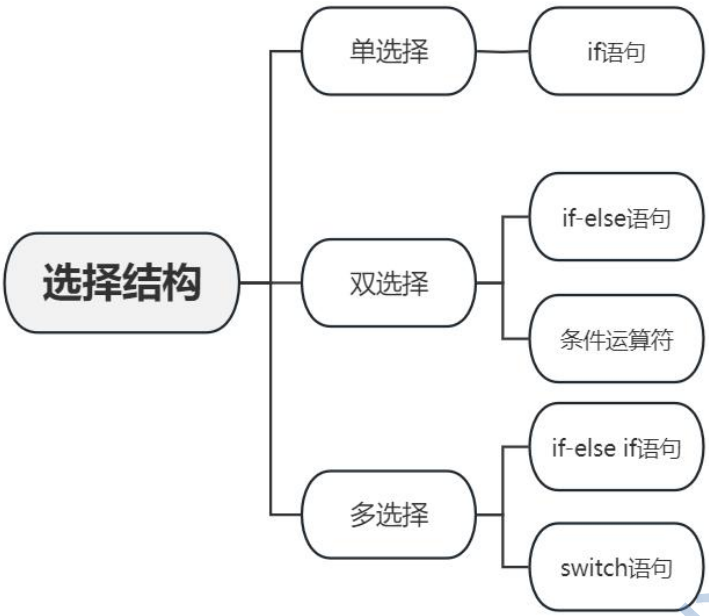
代码执行顺序为：定义变量 time_Data ->键盘输入数据并赋值给 time_Data ->定义变量 hour、minute、second 并初始化为 0-> 提取小时代码->提取分钟代码->提取分秒代码->屏幕输出字符串。

四、选择结构

选择结构（selection structure）是一种条件控制语句，包含一个或

物联星辰计划

多个条件表达式。选择结构的条件语句是让程序能够选择应该执行的代码。



1. if 条件语句

(1)当 if 的判断条件成立时，程序就会执行花括号里的代码；当判断条件不成立时，就不执行花括号内的代码，并结束 if 语句。

单选择通过 if 语句实现，if 语句语法及执行流程如下：

```
if(判断条件)
{
  程序语句块;(判断条件为真时执行)
}
```

例 1：

```c
#include <stdio.h>
int main()
{
    int i = 10;
    if(i == 10)//如果 i=10,则进入 if 语句,执行程序
    {
        printf("欢迎加入物联星辰计划");
        printf("哦耶!");
    }
    return 0;
}
```

例 2：

物联星辰计划

```c
#include <stdio.h>
int main()
{
    int i = 10;
    if (i == 10)
    {
        printf("哦耶!");
    }
    return 0;
}
```

2. if-else 语句

(1)当 if 的判断条件成立时，程序会执行程序语句块 1；当判断条件不成立时，程序会执行程序语句块 2。

双选择通过 if-else 语句实现，if-else 语句的语法及执行流程如下：

```
if (判断条件)
{
  程序语句块 1;（判断条件为真时执行）
}
else
{
  程序语句块 2;（判断条件为假时执行）
}
```

例 3：判断学生分数是否及格

```c
#include <stdio.h>
int main()
{
    int i = 0;
    scanf("%d", &i);//输入学生成绩
    if (i >= 60)//若成绩大于等于60 则输出“及格”
    {
        printf("及格\n");
    }
    else//若成绩小于60 则输出“不及格”
    {
        printf("不及格\n");
    }
    return 0;
}
```

物联星辰计划

3. 条件运算符

条件运算符是 c 语言中唯一一个三目运算符，其求值规则为：如果表达式 1 的值为真，则以表达式 2 的值作为整个条件表达式的值，否则以表达式 3 的值作为整个条件表达式的值，条件表达式通常用于赋值语句之中。
条件运算符的语法及执行流程如下：

```
表达式 1 ? 表达式 2 : 表达式 3
```

例 5：

```c
#include <stdio.h>
int main()
{
    int a = 0,b = 0,max = 0;
    printf("a=");
    scanf("%d",&a);
    printf("b=");
    scanf("%d",&b);
    max = (a>b) ? a : b;
    printf("max = %d",max);
    return 0;
}
```

输入 a=1,b=2;a=6,b=1 时，输出结果如下：
若 a>b 为真，就把 a 赋值给 max；若 a>b 为假，就把 b 赋值给 max。

```
a=1
b=2
max=2
```

```
a=6
b=1
max=6
```

物联星辰计划

4. if-else if 语句

if-else if 语句，是一种多选择的语句，让用户在 if 语句和 else if 语句中选择符合条件表达式的程序语句块，如果以上条件表达式都不符合，就会执行最后的 else 语句。

通过 else if 语句实现的多选择，else if 语句的语法及执行流程如下：

```
if (判断条件 1)
{
   程序语句块 1（判断条件 1 为真时执行）
}
else if (判断条件 2)
{
   程序语句块 2（判断条件 2 为真时执行）
}
```

```
else if (判断条件 3)
{
   程序语句块 3（判断条件 3 为真时执行）
}
 ......
else
{
   程序语句块 n （以上判断条件均为假时执行）
}
```

例 6：年龄分段

```c
#include <stdio.h>
int main()
{
    int age = 0;
    scanf("%d", &age);//输入年龄
    if (age < 18)
    {
        printf("少年\n");
    }
    else if (age >= 18 && age < 30)
    {
        printf("青年\n");
    }
    else if (age >= 30 && age < 50)
    {
        printf("中年\n");
    }
    else if (age >= 50 && age < 80)
    {
        printf("老年\n");
    }
    else
    {
        printf("老寿星\n");
    }
    return 0;
}
```

5. switch 语句

switch 语句也是一种分支语句，常常用于多选择的情况。
switch 语句的语法及执行流程如下：

物联星辰计划

```
switch(常量表达式)
{
    //在一个 switch 中可以有任意数量的 case 语句
    case 常量表达式:
            程序语句块;
}
```

（1）break 语句

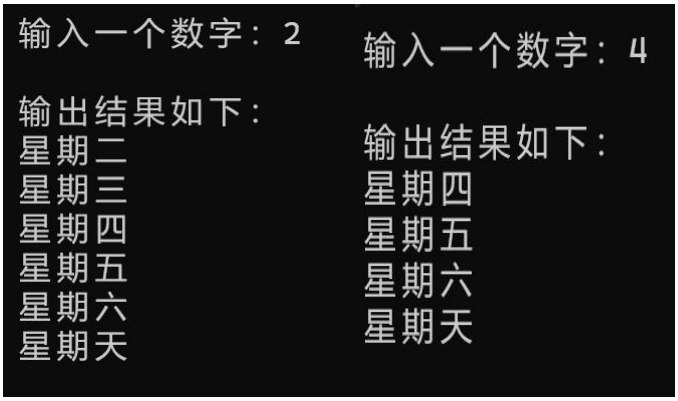
单使用 switch 语句是无法实现分支的，需要搭配 break 语句使用才可以实现真正的分支。

例 7：若没有搭配 break 语句程序如下：

```c
#include <stdio.h>
int main()
{
    int day = 0;
    scanf("%d", &day);
    switch (day)
    {
        case 1:
            printf("星期一\n");
        case 2:
            printf("星期二\n");
        case 3:
            printf("星期三\n");
        case 4:
            printf("星期四\n");
        case 5:
            printf("星期五\n");
        case 6:
            printf("星期六\n");
        case 7:
            printf("星期天\n");
    }
    return 0;
}
```

分别输入 2 和 4，输出结果如下：

物联星辰计划



我们设想的结果是输入“2”，输出“星期二”；输入“4”，输出“星期四”，但是程序输出的结果与设想的结果相差甚远，若搭配 break 就可以解决这个问题。

例 8：搭配 break 语句后的程序如下：

```c
#include <stdio.h>
int main()
{
    int day = 0;
    scanf("%d", &day);
    switch (day)
    {
        case 1:
        {
            printf("星期一\n");
            break;
        }
        case 2:
        {
            printf("星期二\n");
            break;
        }
        case 3:
        {
            printf("星期三\n");
            break;
        }
        case 4:
        {
            printf("星期四\n");
            break;
        }
        case 5:
        {
```

物联星辰计划

```
            printf("星期五\n");
            break;
        }
        case 6:
        {
            printf("星期六\n");
            break;
        }
        case 7:
        {
            printf("星期天\n");
            break;
        }
    }
    return 0;
}
```

分别输入 2 和 4，输出结果如下：

```
输入一个数字：2
输出结果如下：
星期二
```

```
输入一个数字：4
输出结果如下：
星期四
```

由此可见，添加 break 后实际输出结果和我们预想的结果一样。

（2）default 语句

switch 除了搭配 break 语句使用，往往也会搭配 default 语句

当 switch 表达式的值与 case 的值都不匹配时，就会执行 default 语句后的程序语句块。

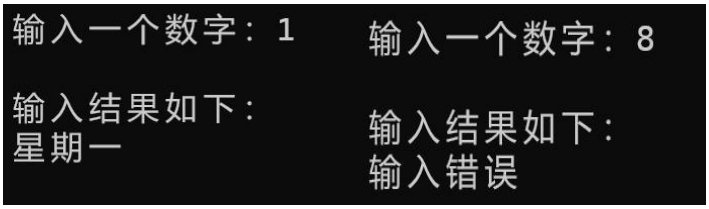
例 9：搭配 break 语句后的程序如下：

```
#include <stdio.h>
int main()
{
    int day = 0;
    scanf("%d", &day);
    switch (day) {
        case 1:
        {
            printf("星期一\n");
            break;
        }
        case 2:
        {
            printf("星期二\n");
```

明珠星愿计划

```
            break;
        }
        case 3:
        {
            printf("星期三\n");
            break;
        }
        case 4:
        {
            printf("星期四\n");
            break;
        }
        case 5:
        {
            printf("星期五\n");
            break;
        }
        case 6:
        {
            printf("星期六\n");
            break;
        }
        case 7:
        {
            printf("星期天\n");
            break;
        }
        default:
        {
            printf("输入错误");
            break;// default 中的break 语句不是必需的
        }
    }
    return 0;
}
```

分别输入 1 和 8，输出结果如下：由于 case 的值没有为 8 的，故执行 default 语句后的程序语句。



物联星辰计划

6. 习题

(1)输入两个整数和一个运算符，输出他们的计算结果(该程序类似一个计算器)。

(2)输入一个字符，如果它是大写字母，就输出“upper”，如果它是小写字母，就输出“lower”，如果它是数字字符，就输出“digit”，如果它是其它字符，就输出“others”。

五、循环结构程序设计

1. 循环结构

循环语句是用于重复执行某条语句（循环体）的语句，它包含三个部分，分别是初始化部分、判断部分和调整循环部分。C 语言提供了 3 中循环语句，分别为 while 语句，do while 语句和 for 语句。

A.for 循环

for 循环形式： for（表达式 1；表达式 2；表达式 3）

表达式 1 为初始化部分，用于初始化循环变量的。

表达式 2 为条件判断部分，用于判断循环时候终止。

表达式 3 为调整部分，用于循环条件的调整。

for 语句的执行过程：

(1) 先求解表达式 1

(2) 求解表达式 2，若其值为真，执行循环体，然后执行下面第(3)步。若为假，则结束循环，转到第(5)步

(3) 求解表达式 3

(4) 转回上面步骤(2)继续执行

(5) 循环结束，执行 for 语句下面的一个语句

B.while 循环：

当 while 循环的表达式为真时，程序进入循环体内执行循环体内的语句。

while 循环形式：

```
while（判断条件）
{
    执行判断语句
}
```

while 循环的特点是：

先判断条件表达式，后执行循环体语句

C.do-while 循环：

do while 循环与 while 循环的语法类似，不同的是 do while 循环至少会执行一次循环体内的内容。

do-while 循环形式：

```
do{语句} while (表达式);
```

do-while 语句的特点：

先无条件地执行循环体，然后判断循环条件是否成立

D.break：

在循环中，只要遇到 break 就停止后期的所有的循环，直接终止循环。

E.continue：

continue 是用于终止本次循环的，也就是本次循环中 continue 后边的代码不会再执行， 而是直接跳转到循环语句的判断部分。进行下一次循环的入口判断。

2. 例题：

输入正数 n，判断 n 是否为素数。若为素数则输出 1，否则输出 0。（提示：素数是指只可以被 1 和其本身整除的正数（1 除外））

六、数组：

1. 数组的概念与定义

数组（Array）是类型相同的数据元素的集合，是 C 语言中的一种构造数据类型，这些元素会顺序地储存在内存的某段区域。数组中的每个元素都有一个序号，这个序号从 0 开始，而不是从我们熟悉的 1 开始，称为下标（Index）。

```
int arr[const];//创建一个数组
//int 是指数组的元素类型
//arr 是指数组名称
//const 是一个常量表达式，用来指定数组的大小
```

在使用数组元素时，指明下标即可。
例如，a[0] 表示第 0 个元素，a[3] 表示第 3 个元素。

2. 数组的初始化

方法一：把第一行的 4 个整数放入数组：

```
a[0]=20;
a[1]=345;
a[2]=700;
a[3]=22;
```

这里的 0、1、2、3 是数组下标，a[0]、a[1]、a[2]、a[3] 是数组元素。
需要注意的是：
1) 数组中每个元素的数据类型必须相同，对于 int a[4];每个元素都必须为 int。

物联星辰计划

2) 数组长度 length 最好是整数或者常量表达式，
3) 访问数组元素时，下标的取值范围为 0 ≤ index < length。

方法二：在定义数组的同时赋值：

```
int a[4] = {20, 345, 700, 22};
```

{ } 中的值即为各元素的初值，各值之间用 , 间隔。
对数组赋初值需要注意以下几点：
不完全初始化。只给部分元素赋初值，当 { } 中值的个数少于元素个数时，只给前面部分元素赋值，剩余元素默认都是 0。如：`int arr[10]={1,2,3};`
2）只能给元素逐个赋值，不能给数组整体赋值。
3）如给全部元素赋值，那么在数组定义时可以不给出数组的长度。
如：`int arr[]={1,2,3,4};`

3. 二维数组的概念与定义

```
dataType arr[length1][length2];
//dataType 为数据类型，arr 为数组名，
//length1 为第一维下标的长度，length2 为第二维下标的长度。
```

我们可以将二维数组看做一个 Excel 表格，有行有列，length1 表示行数，length2 表示列数，要在二维数组中定位某个元素，必须同时指明行和列。
如 `int a[3][4];`
定义了一个 3 行 4 列的二维数组，共有 3×4=12 个元素，数组名为 a，即：
a[0][0], a[0][1], a[0][2], a[0][3]
a[1][0], a[1][1], a[1][2], a[1][3]
a[2][0], a[2][1], a[2][2], a[2][3]
如果想表示第 2 行第 1 列的元素，应该写作 a[2][1]。
也可以将二维数组看成一个坐标系，有 x 轴和 y 轴，要想在一个平面中确定一个点，必须同时知道 x 轴和 y 轴。

在 C 语言中，二维数组是按行排列的。也就是先存放 a[0] 行，再存放 a[1] 行，最后存放 a[2] 行；每行中的 4 个元素也是依次存放。数组 a 为 int 类型，每个元素占用 4 个字节，整个数组共占用 4×(3×4)=48 个字节。
你可以这样认为，二维数组是由多个长度相同的一维数组构成的。

3. 二维数组的初始化

二维数组的初始化可以按行分段赋值，也可按行连续赋值。
例如，对于数组 a[5][3]
按行分段赋值应该写作：

```
int a[5][3]={ {80,75,92}, {61,65,71}, {59,63,70}, {85,87,90},
{76,77,85} };
```

物联星辰计划

按行连续赋值应该写作：

```
int  a[5][3]={80, 75, 92, 61, 65, 71, 59, 63, 70, 85, 87, 90,
76, 77, 85};
```

注意：

1）可以只对部分元素赋值，未赋值的元素自动取“零”值。

如：`int a[3][3] = {{1}, {2}, {3}};`

2）如果对全部元素赋值，那么第一维的长度可以不给出(尽量给出)。

如：`int a[][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};`

5. 对无序数组的查询

所谓无序数组，就是数组元素的排列没有规律。无序数组元素查询的思路也很简单，就是用循环遍历数组中的每个元素，把要查询的值挨个比较一遍。

请看下面的代码：

```
#include <stdio.h>
int main()
{
    int nums[10] = {1, 10, 6, 296, 177, 23, 0, 100, 34, 999};
    int i, num, thisindex = -1;
    printf("Input an integer: ");
    scanf("%d", &num);
    for(i=0; i<10; i++)
    {
        if(nums[i] == num)
        {
            thisindex = i;
            break;
        }
    }
    if(thisindex < 0)
    {
        printf("%d isn't in the array.\n", num);
    }
    else
    {
        printf("%d is in the array, it's index is %d.\n", num,
                                                    thisindex);
    }
```

物联星辰计划

```
    return 0;
}
```

这段代码的作用是让用户输入一个数字，判断该数字是否在数组中，如果在，就打印出下标。

for 循环代码是关键，它会遍历数组中的每个元素，和用户输入的数字进行比较，如果相等就获取它的下标并跳出循环。

注意：数组下标的取值范围是非负数，当 thisindex >= 0 时，该数字在数组中，当 thisindex < 0 时，该数字不在数组中，所以在定义 thisindex 变量时，必须将其初始化为一个负数。

6. 对有序数组的查询

查询无序数组需要遍历数组中的所有元素，而查询有序数组只需要遍历其中一部分元素。例如有一个长度为 10 的整型数组，它所包含的元素按照从小到大的顺序（升序）排列，假设比较到第 4 个元素时发现它的值大于输入的数字，那么剩下的 5 个元素就没必要再比较了，肯定也大于输入的数字，这样就减少了循环的次数，提高了执行效率。

请看下面的代码：

```
#include <stdio.h>
int main()
{
    int nums[10] = {0, 1, 6, 10, 23, 34, 100, 177, 296, 999};
    int i, num, thisindex = -1;

    printf("Input an integer: ");
    scanf("%d", &num);
    for(i=0; i<10; i++)
    {
        if (nums[i] == num)
        {
            thisindex = i;
            break;
        }else if (nums[i] > num)
        {
            break;
        }
    }
    if(thisindex < 0)
    {
```

物联星辰计划

```
        printf("%d isn't  in the array.\n", num);
    }
    else
    {
        printf("%d is  in the array, it's index is %d.\n", num,
                                                    thisindex);
    }
    return 0;
}
```

与前面的代码相比，这段代码的改动很小，只增加了一个判断语句。因为数组元素是升序排列的，所以当 nums[i] > num 时，i 后边的元素也都大于 num 了，num 肯定不在数组中了，就没有必要再继续比较了，终止循环即可。

例题（1）：
有 20 个数按从小到大的顺序存放在一个数组中，从键盘输入一个数，要求使用折半查找法找出该数是数组中第几个元素的值，即输出下标值。如果该数不在数组中，则输出“no found”。
解题思路->折半查找，先对半数组，根据中间值与要查找的值进行大小比较，而后同样进行此条件操作，到最后一个数，若满足中间值=查找值，则找到，若无，则输出 no found。
例题（2）：
输出 15 行的杨辉三角形。

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10  5  1
……
```

解题思路->先是创建了两个变量，一个“s”一个“h”，其中“s”用来临时储存算法运算出来的结果（也就是当前应该输出的值），“h”则是杨辉三角形的高度（也就是需要输出的行数）。
深入研究图形后，我们会看到第 i 行第 j 列的数字是由第 i-1 行第 j 列的数字加上第 i-1 行第 j-1 列的数字构成的。假设数组名称是 array，换成代码表示就是 array[i][j]=a[i-1][j]+a[i-1][j-1]这个式子是二维数组解决杨辉三角的核心。

物联星辰计划